Exhibit II

```
/***************************************************************************/
/*                                                                   */
/*                      connect                                             */
/*                                                                   */
/* Author:  Bruce S. Siegell (bss@buzzard.research.telcordia.com) */
/* File:    connect.c                                              */
/* Date:    Wed Jul 28 10:34:56 EDT 1999                                 */
/*                                                                   */
/* Description:                                                          */
/*    Routines for connecting the monitor near the source to the  */
/*    monitor near the destination.                                      */
/*                                                                   */
/* Copyright (c) 1999 Telcordia Technologies, Inc. (formerly Bellcore). */
/* All rights reserved.                                                  */
/*                                                                   */
/***************************************************************************/

#include <stdio.h>              /* for standard input/output routines.    */
#include <stdlib.h>             /* for atof(), system(), etc.       */
#include <string.h>             /* for strcpy(), etc.               */
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include "ipaware.h"
#include "connect.h"

#define BACKLOG         5       /* the maximum length the queue of  */
                        /* pending connections may grow to. */

/***************************************************************************/
/*                                                                   */
/*    global variables.                                              */
/*                                                                   */
/***************************************************************************/


/***************************************************************************/
/*                                                                   */
/*    module-wide variables - global variables used only in the   */
/*          current file.                                                */
/*                                                                   */
/***************************************************************************/

static int dummy; /* dummy variable.  Not used.                 */


/***************************************************************************/
/*                                                                   */
/*    connect_source - called by the destination monitor to receive     */
/*          a connection from the source monitor.  Returns the    */
/*          socket to be used for the communication.  Returns -1  */
/*          if unsuccessful.                                     */
/*                                                                   */
/***************************************************************************/

int connect_source()
```

```
{
    int lsock;                  /* socket used for listening for    */
                        /* connections.                     */
    int sock;                   /* socket to be used for communication    */
                        /* with the source monitor.         */
    struct sockaddr_in serv_addr;
                        /* information about the server.    */
    struct sockaddr_in cli_addr;
                        /* information about the client.    */
    int clilen;                 /* length of client information.    */

    /* open a TCP socket.                                   */
    if ((lsock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
      fprintf(stderr, "ERROR - can't open stream socket.\n");
      return(-1);
    }

    /* bind an address to the socket so that the client can send to us. */
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(serverport);

    if (bind(lsock, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
      fprintf(stderr, "ERROR - can't bind local address.\n");
      return(-1);
    }

    /* listen for a connection from the client.                 */
    listen(lsock, BACKLOG);

    /* wait for a connection from the client process.           */
    clilen = sizeof(cli_addr);
    sock = accept(lsock, (struct sockaddr *) &cli_addr, &clilen);
    if (sock < 0) {
      fprintf(stderr, "ERROR - accept error.\n");
      return(-1);
    }

    /* we don't need to listen for connections anymore.             */
    close(lsock);

    return sock;
}


/***********************************************************************/
/*                                                             */
/*    connect_destination - called by source monitor to make a     */
/*          connection to the destination monitor.  Returns the    */
/*          socket to be used for the communication or -1 if       */
/*          unsuccessful.                                  */
/*                                                             */
/***********************************************************************/

int connect_destination(char *address)
{
    int sock;                 /* socket to be used for communication    */
                          /* with the destination monitor.    */
    struct sockaddr_in serv_addr;
                          /* information about the server.    */

    /* set up the serv_addr data structure with the information about    */
    /* the server we want to connect to.                    */
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(address);
    serv_addr.sin_port = htons(serverport);

    /* open a TCP socket.                                   */
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
      fprintf(stderr, "ERROR - can't open stream socket.\n");
      return(-1);
    }

    /* connect to the server (the destination monitor).                 */
    if (connect(sock, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
      fprintf(stderr, "ERROR - can't connect to server.\n");
      return(-1);
    }

    return sock;
}


/***************************************************************************/
/*                                                              */
/*    dd2addr - convert an IP address specified in dotted decimal */
/*          notation into an unsigned long in the local host (i.e.,     */
/*          not network) format.                        */
/*                                                              */
/***************************************************************************/

unsigned long dd2addr(char *address)
{
    unsigned long ipaddr;     /* the result.                    */
    int byte;                 /* one byte of the result.        */
    char *buffer;         /* buffer for parsing address.        */
    char *token;          /* token from address string.     */

    ipaddr = 0;

    buffer = strdup(address);

    token = strtok(buffer, ".");
    if (token == NULL) {
      fprintf(stderr,
            "ERROR - Invalid dotted decimal address:  %s.\n",
            address);
      return(-1);
    }
    byte = atoi(token) & 0xff;
    ipaddr = byte << 24;
    token = strtok(0, ".");
    if (token == NULL) {
      fprintf(stderr,
            "ERROR - Invalid dotted decimal address:  %s.\n",
            address);
      return(-1);
    }
    byte = atoi(token) & 0xff;
    ipaddr |= byte << 16;

    token = strtok(0, ".");
    if (token == NULL) {
      fprintf(stderr,
            "ERROR - Invalid dotted decimal address:  %s.\n",
            address);
      return(-1);
    }
    byte = atoi(token) & 0xff;
    ipaddr |= byte << 8;

    token = strtok(0, ".");
    if (token == NULL) {
      fprintf(stderr,
            "ERROR - Invalid dotted decimal address:  %s.\n",
            address);
      return(-1);
    }
    byte = atoi(token) & 0xff;
    ipaddr |= byte;

    return ipaddr;
}
```

```
/**************************************************************************/
/*                                                             */
/*                       connect                               */
/*                                                             */
/* Author:  Bruce S. Siegell (bss@buzzard.research.telcordia.com)                */
/* File:    connect.h                                          */
/* Date:    Wed Jul 28 10:34:56 EDT 1999                             */
/*                                                             */
/* Description:                                                      */
/*    Definitions and function prototypes for connect.               */
/*                                                             */
/* Copyright (c) 1999 Telcordia Technologies, Inc. (formerly Bellcore).
      */
/* All rights reserved.                                              */
/*                                                             */
/**************************************************************************/

#define SERVERPORT      5995  /* the port the server listens on.  */


/**************************************************************************/
/*                                                             */
/*    data structures.                                         */
/*                                                             */
/**************************************************************************/


/**************************************************************************/
/*                                                             */
/*    global variables.                                        */
/*                                                             */
/**************************************************************************/


/**************************************************************************/
/*                                                             */
/*    connect_source - called by the destination monitor to receive     */
/*          a connection from the source monitor.  Returns the    */
/*          socket to be used for the communication.  Returns -1  */
/*          if unsuccessful.                                   */
/*                                                             */
/**************************************************************************/

int connect_source();


/**************************************************************************/
/*                                                             */
/*    connect_destination - called by source monitor to make a    */
/*          connection to the destination monitor.  Returns the   */
/*          socket to be used for the communication or -1 if      */
/*          unsuccessful.                                      */
/*                                                             */
/**************************************************************************/

int connect_destination(char *address);
```